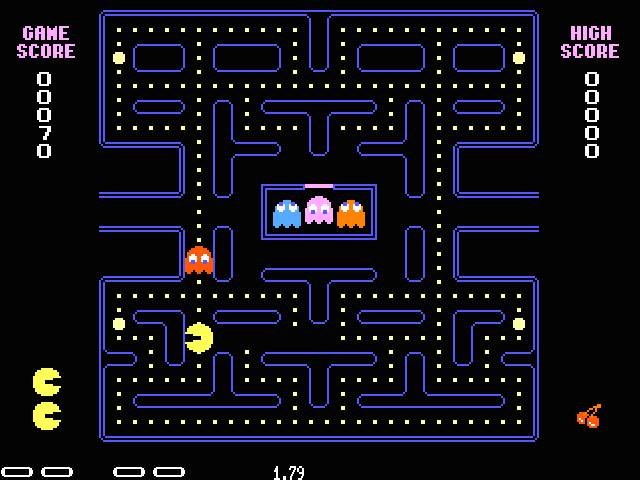
Project Proposal

The aim of this Advanced Higher Computing project is to create a game, using the Pascal programming language, reminiscent of the popular 1980s classic, Pac-Man.



This project will differ in that the game will be a variation of the original Pac-Man, called Hack-Man. Although the maze will work similar to that of the original Pac-Man, in this game it will resemble a microchip. The objective of the game will be to move the player through the maze and collect all “transistors” (point dots or pills), without getting caught by the “anti-viruses” (ghosts).

The game should start from a main menu, with options: "Start Game", "High Scores" and "Help". When the game starts it should load the maze and top high score from file to be displayed on screen. At the end of the game the high scores should be loaded and if the user has a new high score then it should be added. Only the top 5 high scores should be kept.

Timescale

# Estimated



Specification

# Required Algorithms

* 2D array
* Binary search
* Read file
* Write file

# Scope

* There should be a start menu.
* A maze should be displayed (like the Pacman maze).
* The player should move around the maze.
* There should be a score.
* There should be point dots (add to score), ghosts (enemies) and power-ups (let you kill ghosts and give more points than point dots).
* There should be high scores.
* Help documentation should be provided for the user, alongside the game.

# Boundaries

* There does not need to be online help, as help files will be provided with the game.
* Due to the limitations of Pascal, the game should not contain any sound effects.
* Due to the limitations of the Dos window, the graphics will be low quality and represented in text format with colouring.
* There will be no update feature and online DLC will not be provided.
* The maze should only load from one file; multiple mazes will not be supported.
* Only the top ten high scores should be saved and no more.

# Functional Requirements

* A Dos window will display the output.
* The program will run from a Windows exe file.
* There should be a menu at the start with an option to start the game and an option to view high scores.
* The PacMan maze should be loaded from a text file, to allow easy modification during development, to make the code easier to read (without the data embedded in the source code) and to allow for user customisation.
* The maze should be drawn to screen in colour, with the HUD alongside. The HUD should contain the player’s score, lives left and the current high score.
* The user should be able to control the player and move it around screen using WASD in order to eat the point dots in return for points.
* There should be ghosts that chase the player and kill upon contact.
* There should be power ups around the maze that allow the player to turn on the ghosts and have their revenge in return for points.
* The player should get three lives, upon death they should return to the centre and lose a life. Once all lives have been lost the game should end.
* At the end of the game the player should be able to save their score, if it is within the top ten high scores, along with their name.

# Resource Requirements

* Dev Pascal for programming
* Microsoft Word for documentation
* A PC with a keyboard and mouse
* A printer for printing the documentation
* Access to Midlothian file server for storage and file I/O
* Internet access for research purposes

Feasibility Study

# Technical

A Windows computer with Dev Pascal installed is necessary to create this program; the same machine can be used for testing and running. Microsoft Office Word will be used for documentation. The technology is available, as it is provided by the school.

# Economic

This project will cost nothing, as the technology is provided free of charge and the labour is voluntary. Pascal is freeware and any research will be done using materials provided, such as school books and research via the school’s internet connection. These costs are managed by the school and do not concern this project. There should be no economic issues.

# Legal

This open source project is being created for educational purposes and will not be publically distributed. When distributed privately there will be no charge. No personal data is stored within the program or associated files. All code will be created from scratch and no module libraries shall be used. There should be no conflicts with copyright or the law.

# Schedule

A timescale is given earlier on in this document; it shows that it should be possible to complete the project on time. The deadline for the completed programs and all documentation is the end of February 2013. This should allow ample time for development and the project should be completed before it is due. Issues which may affect this are holidays throughout the year and smaller projects that will be done simultaneously to this one, but given good time keeping this project should go to schedule.

Altogether, the project should be feasible.

Pseudocode Plan (tmp)

**Main**

Repeat forever  
  
 *// Menu* Draw menu title  
 **Draw menu options**, with first option selected  
 **Handle input**, redraw menu options (in loop)  
 Clear Screen  
  
 *// Game* **Load maze from file**  
 **Find sprites in maze**  
 **Draw maze**  
 Draw HUD  
 Setup final game settings  
 **Start the game**  
 *// Game has ended, highscores* Clear the screen  
 Unload game settings, setup for menu again  
 Handle new highscore  
 Show highscores  
  
 *// Will start at the menu again*  
End repeat

**Draw menu options**

If menuOption = 1  
 display “> Start Game <”  
Else  
 display “ Start Game “  
End if  
If menuOption = 2  
 display “> High Scores <”  
Else  
 display “ High Scores “  
End if  
If menuOption = 3  
 display “> Help <”  
Else  
 display “ Help “  
End if

**Handle menu input**

Get key input from user  
If key = w, W or UP  
 *// up pressed*  
 menuOption -= 1  
 if menuOption < 1  
 menuOption = 3  
Else if key = s, S or DOWN  
 *// down pressed*  
 menuOption += 1  
 if menuOption > 3  
 menuOption = 1  
Else If key = SPACEBAR or RETURN  
 *// select key pressed*  
 if menuOption = 1  
 inMainMenu = false  
 else if menuOption = 2  
 show highscores  
 else if menuOption = 3  
 show help  
**Draw menu options**

**Load maze from file**

Setup file for loading  
while not( end of file ) and lineNumber < 23  
 Read a line of the file  
 // now parse the line  
 For charCounter = 1 to 53  
 **Substitute character for integer that represents cell**  
 Add integer to maze 2D array  
 Next charCounter  
 lineNumber += 1  
End while  
Close file

**Characters and integers that represent cells**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Empty | Wall | Coin | Fruit | Powerup | Player  Spawn | Teleport | Ghost Home | Ghost  Spawn | Ghost  Door |
| File  char | . or space | X | \* | ? | @ | P | T | G | g | H |
| Code integer | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Draw  char | \_ | \_ | o | ó | О | ☻ | \_ | ± | \_ | \_ |

**Find sprites in maze**

ghostHomesFound = 0  
coinsFound = 0  
*// linear search through maze*  
For column = 1 to 53  
 For row = 1 to 23  
 cell = maze[column, row]  
 If cell is player spawn  
 Set playerSpawn coordinate  
 Else if cell is ghost spawn  
 Set ghostSpawn coordinate  
 Else if cell is fruit spawn  
 Set fruitSpawn coordinate  
 Else if cell is coin  
 coinCount += 1  
 Else if cell is ghost home  
 // found 1, 3 to find  
 ghostHomesFound += 1  
 Set ghost home number ghostHomesFound coordinate  
 End if  
 Next row  
Next column

**Draw maze**

For column = 1 to 53  
 For row = 1 to 23  
 cell = maze[column, row]  
 **Draw cell**  
 Next row  
Next column

**Draw cell**

Set text colour to white  
Set text background colour to black  
Draw character for integer that represents cell

**Start the game**

Setup variables and other stuff for play  
Wait 1.5 seconds before starting game  
  
*// Game / pause menu loop*  
Repeat  
 *// Game clock loop*  
 Repeat  
  
 Get key input without blocking  
 Work out next direction to move player  
  
 *// spawn fruit every 2000 ticks (20 secs)*  
 If fruitTicks > 2000  
 Spawn in the fruit  
 fruitTicks = 1  
  
 *// deshockify ghosts after 800 ticks (8 secs)*  
 If shockTicks > 800  
 Set ghosts to not shocked  
 shockTicks = 1  
 End if  
  
 *// move the player  
 // every 10 ticks (100ms) if moving horizontal  
 // every 20 ticks (200ms) if moving vertical*  
 If (direction is horizontal and playerTicks > 10) or (direction is vertical and playerTicks > 20)  
 **Move player**  
 playerTicks = 1  
 End If  
   
 *// move the ghosts  
 // every 15 ticks (150ms) if moving horizontal  
 // every 25 ticks (250ms) if moving vertical*  
 For ghost = 1 to 4  
 If ghost is alive  
 If (direction horizontal and ghost’s ticks > 15) or (vertical and ghost’s ticks > 25)  
 Move the ghost  
 Ghost’s ticks = 1  
 End if  
 End if  
 Next ghost  
   
 *// spawn the ghosts every 200 ticks (2 secs)* If ghostSpawnTicks > 200  
 For ghost = 1 to 4  
 If ghost is alive  
 Spawn the ghost  
 End if  
 Next ghost  
 ghostSpawnTicks = 1  
 End if  
  
 *// handle ghost hitting player* For ghost = 1 to 4  
 If (player location = ghost location) and ghost is alive  
 If ghost not shocked  
 Player die animation  
 Reset Sprites  
 lives -= 1  
 Break for  
 Else  
 Player eat ghost animation  
 score += 200  
 Set ghost alive = false  
 End if  
 End if  
 Next ghost  
   
 *// check for a level complete* If coins left < 1  
 Level complete animation  
 Reset board  
 level += 1  
 Increase player speed  
 Increase ghosts’ speeds  
  
 Update the HUD  
   
 *// check for a game over* If lives < 1  
 Break game / pause menu loop  
  
 *// handle clock timer* Pause for 10ms (equivelant to 1 ‘tick’)  
 playerTicks += 1  
 SpawnTicks += 1  
 Increment each ghost’s ticks by 1  
 If fruit not currently spawned  
 fruitTicks += 1  
 If ghosts currently shocked  
 shockTicks == 1  
  
 Until key pressed = “Esc”  
  
 Show pause options  
 Get key input  
 If key = “Q”  
 Break game / pause menu loop  
  
Next repeat

**Move Player**

*// remove old player char*  
Move cursor to old player location  
Draw empty space (see table above)  
  
*// check if chosen next direction is valid*  
If **nextDirection is valid move direction**  
 direction = nextDirection  
End if  
  
*// change the player’s position*  
If **direction is valid move direction**  
   
 *// get new coordinates*  
 **Move ‘playerLocation’ cell in direction ‘direction’**  
   
 *// check for obstacles*  
 cell = maze[playerLocation.x, playerLocation.y]  
 If cell is coin  
 coinCount -= 1  
 score += 10  
 Replace cell with empty space  
 Else if cell is fruit  
 score += 100  
 Replace cell with empty space  
 Start fruit spawner again  
 Else if cell is powerup  
 score += 50  
 Set ghosts to shocked  
 Start de-shockify timer  
 Got powerup animation  
 Replace cell with empty space  
 Else if cell is teleport  
 **Teleport playerLoc to other side**  
 End if  
  
End if  
  
*// redraw the player*  
Move cursor to new player location  
Draw the player (see table above)

**Is direction valid move direction**

valid = false **Get next location for direction and current location**  
If direction is vertical  
 *// check if in 3 wide column*  
 If (cell is not wall) and (cell to left is not wall) and (cell to right is not wall)  
 valid = true  
 End if  
If direction is horizontal  
 If (cell is not wall) and (cell after that is not wall)  
 valid = true  
 End if  
End if  
  
*// check for out of bounds locations*  
If (column > 53 or < 0) and (row > 23 or < 0)  
 valid = false

**Move location in direction**

Case direction of  
 1 : location.x += 1  
 2 : location.y += 1  
 3 : location.x -= 1  
 4 : location.y -= 1  
End case

**Teleport location**

If location column = 1  
 location column = 53  
Else if location column = 53  
 location column = 1

Code Outline

**Main**

Repeat forever  
 Do menu - Draw Menu  
 - Handle input, redraw menu, in loop  
 Do game  
 - **Load maze from file and setup**  
 - **Draw maze and HUD**  
 - **Setup final game settings**  
 - **Start the game**  
Game has ended, handle highscores  
Will start at the menu again (due to repeat)  
End repeat

**Load maze from file and setup**

Setup file for loading  
Loop through lines  
 Read a line of the file  
 Parse the line and add to maze 2D array  
 - **Substitute char for an integer that represents cell** (ie wall, powerup, spawn)  
 - See “File char” and “Code Integer” rows from table below  
 - This means several chars can represent one cell  
Close file

* Maze file has characters from “File Char” row in table below in a grid 53x23 to represent cells that will be drawn. The file should be called “maze.txt” and it should be located in the program’s root folder and be in plain text format.
* Each horizontal passage should be one cell high.
* Each vertical passage should be 3 cells wide (this is to compensate for characters being taller than they are wide).
* Teleport cells can only be in column 1 or 53. Upon player or ghost contact they will cause the player or ghost to move to the cell on the opposite side horizontally, eg column 53 to column 1 or column 1 to column 53, with the row number remaining the same. The landing cell should be empty and available for both player and ghost access.
* There must be only one player spawn, exactly three ghost homes and only one ghost spawn.

**Characters and integers that represent cells**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Empty | Wall | Coin | Fruit | Powerup | Player  Spawn | Teleport | Ghost Home | Ghost  Spawn | Ghost  Door |
| File  Char | . or space | X | \* | ? | @ | P | T | G | g | H |
| Code Integer | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Draw  Char | \_ | \_ | o | ó | О | ☻ | \_ | ± | \_ | \_ |

**Draw maze**

Loop 1 to 53 for columns  
 Loop 1 to 23 for rows  
 Draw cell  
 - Get cell from 2D maze array (column, row)  
 - Use integer that represents cell to draw character for that cell  
 - See “Code Integer” and “Draw Char” rows of table above  
 Next row  
Next column

**Draw HUD**

* The HUD should be vertical along the right side of the maze
* Should contain:
  + High score (5 digits long)
  + Game score (5 digits long)
  + Level (single number)
  + Lives (3 pacman characters, represent a life each)
* Beneath the maze should be a line that can have any textual output, eg debug text and pause menu.

**Setup final game settings**

Take the player out of the maze  
- The code later on will spawn the player in at the recorded player spawn location  
Take the fruit out of the maze  
- The code later on will periodically spawn in the fruit at the recorded fruit spawn location  
Initialise variables for start game  
- Current player location, directions, ghost data, score, lives, level, fruit spawner, clock tick variables, ghost shock timer variables, current key pressed  
Initialise the keyboard so that the non-blocking key input can be used

**Start the game**

Setup variables and other stuff for play  
Wait 1.5 seconds before starting game  
  
Game / pause menu loop  
 Game clock loop  
  
 Get key input without blocking  
 Work out next direction to move player  
Spawn fruit every 2000 ticks (20 secs)  
Deshockify ghosts after 800 ticks (8 secs)  
**Move the player**  
- Every 10 ticks (100ms) if moving horizontal- Every 20 ticks (200ms) if moving vertical  
 Move the ghosts- Every 15 ticks (150ms) if moving horizontal- Every 25 ticks (250ms) if moving vertical  
 Spawn the ghosts every 200 ticks (2 secs)  
 Handle ghost hitting player  
 - If ghost shocked, player die  
 - Else ghost die  
 Check for a level complete  
 - If so, then end level and increase speeds  
 Update the HUD  
 Check for a game over, break loops if necessary  
 Handle clock timer - Pause for 10ms (equivelant to 1 ‘tick’)  
 - Increase relevant tick counter variables  
  
 Exit game clock loop on Escape key press  
  
 Show pause options  
 Get key input  
 - If key = Q, break game / pause menu loop

**Move Player**

Remove old player char  
Check if chosen next direction is valid  
- If true, then set current direction = next direction  
**If current direction is valid move direction**  
 Move ‘playerLocation’ cell in direction ‘direction’  
 Check for obstacles  
 - For a coin add 10 points and decrease coin count  
 - For a fruit add 100 points and restart fruit spawner  
 - For a powerup add 50 points and make the ghosts shocked  
 - For a teleport move player horizontally to other side  
Redraw the player (“Player Spawn” character from “Draw Char” row in table above)

**Is direction valid move direction**

**Get next location for direction and current location**Check for valid vertical direction  
 - Need to handle 3 cells wide vertical passage  
 - Check next location cell and the cells on its left and right  
Check for valid horizontal direction  
 - Need to stop with a one cell gap to the left or right before a wall  
 - Check next location cell and the cell after that  
Check for invalid out of bounds locations

Pseudocode

# Main

1. Repeat forever2. Draw menu title  
**3**. **Draw menu options**, menuOption = 1  
**4**. **Handle input, redraw menu**  
5. Clear Screen**6**. **Load maze from file**  
**7**. **Find sprites in maze**  
8. **Draw maze**  
9. Draw HUD  
10. Setup final game settings  
**11**. **Start the game**12. Clear the screen  
13. Unload game settings, setup for menu again  
14. Handle new highscore  
15. Show highscores  
16. End repeat

# 3

3.1. If menuOption = 1  
3.2. display “> Start Game <”  
3.3. Else  
3.4. display “ Start Game “  
3.5. End if  
3.6. If menuOption = 2  
3.7. display “> High Scores <”  
3.8. Else  
3.9. display “ High Scores “  
3.10. End if  
3.11. If menuOption = 3  
3.12. display “> Help <”  
3.13. Else  
3.14. display “ Help “  
3.15. End if

# 4

4.1. Get key input from user  
4.2. If key = w, W or UP  
4.3. menuOption -= 1  
4.4. If menuOption < 1  
4.5. menuOption = 3  
4.6. Else if key = s, S or DOWN  
4.7. menuOption += 1  
4.8. If menuOption > 3  
4.9. menuOption = 1  
4.10. Else If key = SPACEBAR or RETURN  
4.11. If menuOption = 1  
4.12. inMainMenu = false  
4.13. Else if menuOption = 2  
4.14. Show highscores  
4.15. Else if menuOption = 3  
4.16. Show help  
4.17. End if  
4.18. End if  
4.19. **Draw menu options** (step **3**)

# 6

6.1. Setup file for loading  
6.2. while not( end of file ) and lineNumber < 23  
6.3. Read a line of the file  
6.4. For charCounter = 1 to 53  
6.5. Substitute character for integer that represents cell  
6.6. Add integer to maze 2D array  
6.7. Next charCounter  
6.8. lineNumber += 1  
6.9. End while  
6.10. Close file

# 7

7.1. Integer ghostHomesFound = 0  
7.2. Integer coinsFound = 0  
7.3. For column = 1 to 53  
7.4. For row = 1 to 23  
7.5. cell = maze[column, row]  
7.6. If cell is player spawn  
7.7. Set playerSpawn coordinate  
7.8. Else if cell is ghost spawn  
7.9. Set ghostSpawn coordinate  
7.10. Else if cell is fruit spawn  
7.11. Set fruitSpawn coordinate  
7.12. Else if cell is coin  
7.13. coinCount += 1  
7.14. Else if cell is ghost home  
7.15. ghostHomesFound += 1  
7.16. Set ghost home number ghostHomesFound coordinate  
7.17. End if  
7.18. Next row  
7.19. Next column

# 8

8.1. For column = 1 to 53  
8.2. For row = 1 to 23  
8.3. cell = maze[column, row]  
**8.4**. **Draw cell**  
8.5. Next row  
8.6. Next column

# 8.4

8.4.1. Set text colour to white  
8.4.2. Set text background colour to black  
8.4.3. Draw character for integer that represents cell

# 11

**11.1**. **Setup variables and other stuff for play**  
11.2. Wait 1.5 seconds before starting game  
11.3.Repeat  
11.4. Repeat  
11.5. Get key input without blocking  
**11.6**. **Work out next direction to move player**  
11.7. If fruitTicks > 2000  
11.8. Spawn in the fruit  
11.9. fruitTicks = 1  
11.10. If shockTicks > 800  
11.11. Set ghosts to not shocked  
11.12. shockTicks = 1  
11.13. End if  
11.14. If (direction is horizontal and playerTicks > 10) or (direction is vertical and playerTicks > 20)  
**11.15**. **Move player**  
11.16. playerTicks = 1  
11.17. End If  
11.18. For ghost = 1 to 4  
11.19. If ghost is alive  
11.20. If (direction horizontal and ghost’s ticks > 15) or (vertical and ghost’s ticks > 25)  
**11.21**. **Move the ghost**  
11.22. Ghost’s ticks = 1  
11.23. End if  
11.24. End if  
11.25. Next ghost   
11.26. If ghostSpawnTicks > 200  
11.27. For ghost = 1 to 4  
11.28. If ghost is alive  
**11.29**. **Spawn the ghost**  
11.30. End if  
11.31. Next ghost  
11.32. ghostSpawnTicks = 1  
11.33. End if  
11.34. For ghost = 1 to 4  
11.35. If (player location = ghost location) and ghost is alive  
11.36. If ghost not shocked  
11.37. Player die animation  
**11.38**. **Reset Sprites**  
11.39. lives -= 1  
11.40. Break for  
11.41. Else  
11.42. Player eat ghost animation  
11.43. score += 200  
11.44. Set ghost alive = false  
11.45. End if  
11.46. End if  
11.47. Next ghost  
11.48. If coins left < 1  
11.49. Level complete animation  
11.50. Reset board  
11.51. level += 1  
**11.52**. **Increase player speed**  
**11.53**. **Increase ghosts’ speeds**  
11.54. End if  
11.55. Update the HUD  
11.56. If lives < 1  
11.57. Break game / pause menu loop  
11.58. Pause for 10ms (equivelant to 1 ‘tick’)  
11.59. playerTicks += 1  
11.60. SpawnTicks += 1  
11.61. Increment each ghost’s ticks by 1  
11.62. If fruit not currently spawned  
11.63. fruitTicks += 1  
11.64. If ghosts currently shocked  
11.65. shockTicks == 1  
11.66. Until key pressed = “Esc”  
11.67. Show pause options  
11.68. Get key input  
11.69. If key = “Q”  
11.70. Break game / pause menu loop  
11.71. Next repeat

# 11.1

11.1.1. direction = 1  
11.1.2. nextDirection = 1  
11.1.3. Coordinate playerLocation = player spawn location  
11.1.4. For ghost = 1 to 4  
11.1.5. Ghost direction = 1 or 3 (random)  
11.1.6. Ghost ticks = 1  
11.1.7. Ghost alive = false  
11.1.8. Ghost shocked = false  
11.1.9. Next ghost  
11.1.10. Ghost 1 location = ghost spawn location  
11.1.11. Ghost 2 location = ghost home 1 location  
11.1.12. Ghost 3 location = ghost home 2 location  
11.1.13. Ghost 4 location = ghost home 3 location  
11.1.14. Score = 0  
11.1.15. Lives = 3  
11.1.16. Level = 1  
11.1.17. playerMoveWait = 10  
11.1.18. ghostMoveWait = 15  
11.1.19. playerTicks = 1  
11.1.20. fruitTicks = 1  
11.1.21. shockTicks = 1  
11.1.22. ghostSpawnTicks = 999  
11.1.23. Initialise keyboard for “keyboard” module

# 11.6

11.6.1. if key = 'd' or 'D'  
11.6.2. nextDirection = 1  
11.6.3. else if key = 's' or 'S'  
11.6.4. nextDirection = 2  
11.6.5. else if key = 'a' or 'A'  
11.6.6. nextDirection = 3  
11.6.7. else if key = 'w' or 'W'  
11.6.8. nextDirection = 4  
11.6.9. End if

# 11.15

11.15.1. Move cursor to old player location  
11.15.2. **Draw empty space** (step **8.4**)  
**11.15.3**. If **nextDirection is valid move direction**  
11.15.4. direction = nextDirection  
11.15.5. If **direction is valid move direction** (step **11.15.3**)  
**11.15.6**. **Move ‘playerLocation’ cell in direction ‘direction’**  
11.15.7. cell = maze[playerLocation.x, playerLocation.y]  
11.15.8. If cell is coin  
11.15.9. coinCount -= 1  
11.15.10. score += 10  
11.15.11. Replace cell with empty space  
11.15.12. Else if cell is fruit  
11.15.13. score += 100  
11.15.14. Replace cell with empty space  
11.15.15. Start fruit spawner again  
11.15.16. Else if cell is powerup  
11.15.17. score += 50  
11.15.18. Set ghosts to shocked  
11.15.19. Start de-shockify timer  
11.15.20. Got powerup animation  
11.15.21. Replace cell with empty space  
11.15.22. Else if cell is teleport  
**11.15.23**. **Teleport playerLoc to other side**  
11.15.24. End if  
11.15.25. End if  
11.15.26. Move cursor to new player location  
11.15.27. **Draw the player character** (step **8.4**)

# 11.15.3

11.15.3.1. valid = false11.15.3.2. **Get next location for direction and current location** (step **11.15.6**)11.15.3.3. If direction is vertical  
11.15.3.4. If (cell is not wall) and (cell to left is not wall) and (cell to right is not wall)  
11.15.3.5. valid = true  
11.15.3.6. End if  
11.15.3.7. If direction is horizontal  
11.15.3.8. If (cell is not wall) and (cell after that is not wall)  
11.15.3.9. valid = true  
11.15.3.10. End if  
11.15.3.11. End if  
11.15.3.12. If (column > 53 or < 0) and (row > 23 or < 0)  
11.15.3.13. valid = false

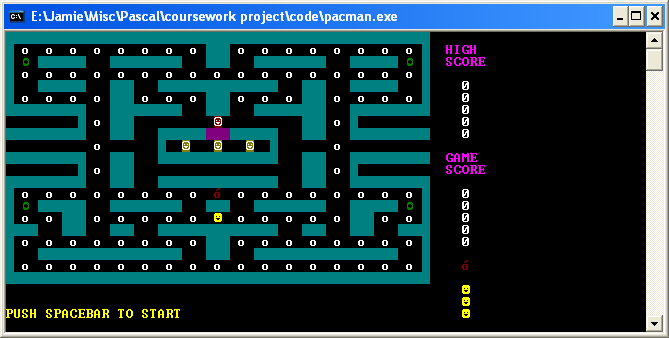
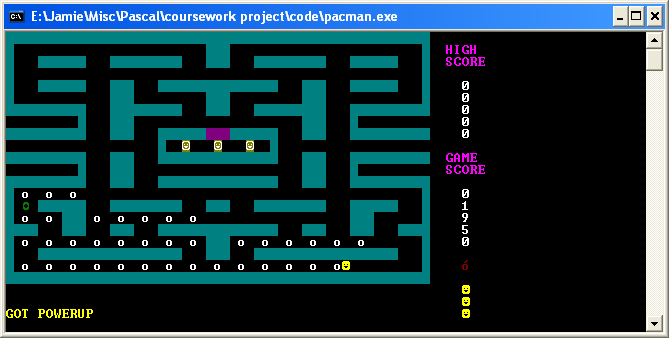
# 11.15.6

11.15.6.1. Case direction of  
11.15.6.2. 1 : location.x += 1  
11.15.6.3. 2 : location.y += 1  
11.15.6.4. 3 : location.x -= 1  
11.15.6.5. 4 : location.y -= 1  
11.15.6.6. End case

# 11.15.23

11.15.23.1. If location column = 1  
11.15.23.2. location column = 53  
11.15.23.3. Else if location column = 53  
11.15.23.4. location column = 1

Prototype

Implementation

